National Institute of
Standards and Technology
U.S. Department of Commerce

**NIST Interagency Report 7693
(Draft)**

Specification for Asset Identification 1.1 (Draft)

John Wunder
Adam Halbardier
David Waltermire

NIST Interagency Report 7693 (Draft)

Specification for Asset Identification 1.1 (Draft)

John Wunder
Adam Halbardier
David Waltermire

C O M P U T E R S E C U R I T Y

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

December 2010

U.S. Department of Commerce

Gary Locke, Secretary

National Institute of Standards and Technology

Dr. Patrick D. Gallagher, Director

Reports on Computer Systems Technology

National Institute of Standards and Technology Interagency Report 7693 (Draft)
32 pages (Dec. 2010)

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

Acknowledgments

The authors, John Wunder of The MITRE Corporation, Adam Halbardier of Booz Allen Hamilton, and David Waltermire of the National Institute of Standards and Technology (NIST), wish to thank their colleagues who reviewed drafts of this document and contributed to its technical content. The authors would like to acknowledge Harold Booth of NIST, Karen Scarfone of G2, Inc., Joseph Wolfkiel of the Defense Information Systems Agency (DISA), Jim Ronayne of the Department of Defense Computer Network Defense Research and Technology Program Management Office (DoD CND R&T PMO), Gary Newman of Belarc, Gerard McGuire of The MITRE Corporation, and Mark Johnson and Paul Cichonski of Booz Allen Hamilton for their keen and insightful assistance throughout the development of the document.

Abstract

Asset identification plays an important role in an organization's ability to quickly correlate different sets of information about assets. This specification provides the necessary constructs to uniquely identify assets based on known identifiers and/or known information about the assets. This specification describes the purpose of asset identification, a data model for identifying assets, methods for identifying assets, and guidance on how to use asset identification. It also identifies a number of known use cases for asset identification.

Feedback

NIST welcomes feedback on the Asset Identification specification. Please submit public comments to emerging-specs@nist.gov or private comments to asset-reporting-comments@nist.gov.

Trademark Information

Windows XP is a registered trademark of Microsoft Corporation. CPE is a trademark of MITRE Corporation.

All other registered trademarks or trademarks belong to their respective organizations.

Table of Contents

List of Figures

List of Tables

1. Introduction

One of the primary requirements for performing asset management is the ability to identify assets based on some set of data known about them. Asset identification, the ability to uniquely identify an asset, allows for correlation of data across multiple sources, reporting of asset information across different organizations and databases, targeted actions against specific assets, and usage of asset data in other business processes.

Unfortunately, neither a unified method nor a published specification for performing asset identification exists at this time. Existing security automation specifications either do not consider asset identification or represent identification information differently than other specifications with which they interoperate. This means that correlation of data relies on a transformation process between each specification, which is expensive and unreliable. Creation of such a unified method and specification for performing asset identification would allow for greater interoperability, increased capabilities, and easier implementation of asset management processes.

This Asset Identification specification describes how asset management processes and other specifications may identify assets using some set of information known or generated about the asset. It describes the data model and representation of asset identification information and it provides requirements for consuming and producing identification information. Representation of asset information, requirements for usage of asset information, and requirements for how the information that identifies assets is collected or generated are out of scope for this specification.

For the purposes of this specification, an asset is considered to be anything that has value to an organization. For example, computing devices are one form of asset that many organizations track. This specification, however, does not limit asset identification to identifying computing devices; any type of asset may be identified. The specification itself provides constructs for identifying many types of assets, and users may extend the model to include other asset types if they wish to identify asset types that are not addressed in the specification.

It is expected that other standards, data formats, tools, processes, and organizations will reference this specification to describe how to represent asset identification information. This will ensure compatibility of asset identifications among these components and allow for improved asset management processes.

While this specification was developed to support the immediate needs of the security automation community, it is expected that it will be valuable in general asset management processes both inside and outside of the security automation space.

1.1 Purpose and Scope

The purpose of this document is to define the Asset Identification specification, a standardized model for representing and identifying assets.

The scope of this document is to give an introduction to Asset Identification, give guidelines on using Asset Identification, describe the Asset Identification data model, and document conformance requirements to comply with Asset Identification. Other versions of Asset Identification and the associated component specifications, including emerging specifications and future versions, are not addressed here.

Future versions of Asset Identification will be defined in distinct revisions of this document, each clearly labeled with a document revision number and the appropriate Asset Identification version number.

1.2 Audience

This specification is intended for authors of specifications that must support asset identifications, implementers of those specifications, system integrators composing architectures from tools that implement those specifications, and end users who wish to understand how these tools work.

1.3 Document Structure

The remainder of this document is organized into the following major sections:

- Section 2 defines the terms used within this specification and provides a list of common abbreviations.
- Section 3 describes how this specification fits with related specifications.
- Section 4 defines the conformance requirements for asset identification.
- Section 5 gives an overview of asset identification.
- Section 6 describes the asset identification data model constructs.
- Appendix A describes possible use cases for asset identification.
- Appendix B documents the normative references for this specification

1.4 Document Conventions

Throughout this specification, whenever a specific term from the data model is referenced, as defined in Section 6, the term is written in `Courier New` font. When referencing a specification listed in Appendix B, the name will be written between brackets, such as [XML Schema].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119].

Both inline and indented forms use qualified names to refer to specific XML elements. A qualified name associates a named element with a namespace. The namespace identifies the specific XML schema that defines (and consequently may be used to validate) the syntax of the element instance. A qualified name declares this schema to element association using the format '*prefix*:*element-name*'. The association of prefix to namespace is defined in the metadata of an XML document and generally will vary from document to document. In this specification, the conventional mappings listed in Table 1-1 are used.

Table 1-1: Conventional XML Mappings

Mappings Prefix	Namespace URI	Schema
ai	http://scap.nist.gov/schema/asset-identification/1.1	Asset Identification 1.1
core	http://scap.nist.gov/schema/reporting-core/1.1	SCAP Reporting Core 1.1
xal	urn:oasis:names:tc:ciq:xsdschema:xAL:2.0	OASIS extensible Address Language
xnl	urn:oasis:names:tc:ciq:xsdschema:xNL:2.0	OASIS extensible Name Language

2. Terms and Abbreviations

2.1 Terms

This section defines a set of common terms used within the document.

Asset: Anything that has value to an organization, including, but not limited to, another organization, person, computing device, information technology (IT) system, IT network, IT circuit, software (both an installed instance and a physical instance), virtual computing platform (common in cloud and virtualized computing), and related hardware (e.g., locks, cabinets, keyboards).

Asset Identification: The use of attributes and methods to establish and maintain unique information about a given asset.

Asset Identification Element: A complete expression of an asset identification using the constructs defined in this specification expressed in a binding.

Computing Device: A machine (real or virtual) for performing calculations automatically.

Extension Identifier: Any piece of identifying information provided in an Asset Identification Element that is not explicitly defined in the Asset Identification schema.

Identifying Information: The set of an asset's attributes that may be useful for identifying that asset, including discoverable information about the asset and identifiers assigned to the asset.

Matching: The process of determining whether two or more asset identification expressions refer to the same asset.

Network: An information system(s) implemented with a collection of interconnected components. Such components may include routers, hubs, cabling, telecommunications controllers, key distribution centers, and technical control devices.

Organization: An entity of any size, complexity, or positioning within an organizational structure (e.g., a federal agency, or, as appropriate, any of its operational elements).

Relationship Identifier: Identifying information where the value is a relationship to another asset.

Software: Computer programs and associated data that may be dynamically written or modified during execution.

System: A discrete set of information resources organized for the collection, processing, maintenance, use, sharing, dissemination, or disposition of information.

Synthetic Identifier: An identifier that is assigned to an asset in the context of some management domain.

2.2 Acronyms

BIOS	Basic Input/Output System
CIDR	Classless Inter-Domain Routing
CPE	Common Platform Enumeration

FQDN	Fully-Qualified Domain Name
GUID	Globally Unique Identifier
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IT	Information Technology
ITL	Information Technology Laboratory
MAC	Media Access Control
NIST	National Institute of Standards and Technology
OASIS	Organization for the Advancement of Structured Information Standards
RFC	Request for Comment
SMBIOS	System Management BIOS
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
xAL	extensible Address Language
XML	Extensible Markup Language
xNL	extensible Naming Language
XSD	XML Schema

3. Relationship to Existing Standards and Specifications

This specification defines the constructs and methods for representing asset identification information and thus can be leveraged by any other specification where identifying assets is required or beneficial.

This specification uses several industry-standard mechanisms for representing identification information and providing conformance requirements.

Common Platform Enumeration (CPE)™ is a structured naming scheme for information technology systems, platforms, and packages. Based upon the generic syntax for Uniform Resource Identifiers (URI), CPE includes a formal name format. CPE version 2.3 well-formed names are used as software-identifying information by this specification.

The extensible Address Language (xAL) by the Organization for the Advancement of Structured Information Standards (OASIS) is an XML standard format for representing international address information. Asset Identification leverages xAL to represent address information for assets.

The extensible Name Language (xNL) by OASIS is an XML standard format for representing the names of people and organizations. Asset Identification leverages xNL to represent the names of people and organizations.

4. Conformance

A product may want to claim conformance with this specification so that users and organizations can use the product with the assurance that the product can identify assets in a consistent and standard manner. The ability for a product to identify assets in a standard manner increases the likelihood of interoperability between conforming products. This section defines the criteria for products to claim conformance with this specification.

4.1 Product Conformance

Products are divided into two roles based on their use of asset identification information: consumers and producers.

- Consuming products (“consumers”) must be able to receive and understand information in compliance with this specification.
- Producing products (“producers”) must create asset identification information in a format compliant with this specification.

A product may be both a consumer and producer. The following subsections document the conformance requirements for the two types of products.

4.1.1 Consumers

Any consuming product claiming conformance to this specification MUST adhere to the following requirements.

- The consumer SHALL be capable of processing the identification information represented in constructs consistent with the Asset Identification data model without error. **REF:** Section 6
- The consumer MAY attempt to consume constructs that are invalid per the Asset Identification data model. **REF:** Section 6
- The consumer MAY consume extension identifiers and use them as an input into a matching process. **REF:** Section 5.2.4

4.1.2 Producers

Any producing product claiming conformance to this specification MUST adhere to the following requirements.

- The producer SHALL accurately produce the asset identification element in XML consistent with the data model. **REF:** Section 6
- When representing identification information, the producer SHOULD provide as much information as is sufficient to allow for a match. **REF:** Section 5.3
- When representing identification information, the producer MAY provide as much or as little identifying information as allowed in the data model per other recommendations or tool capabilities. **REF:** Section 5.3
- The producer MAY provide extension identifiers for any asset identification element. **REF**: Section 5.2.4

5. Asset Identification Overview

This section gives an overview of Asset Identification and its key concepts.

5.1 Scope

In order to support the variety of use c**ase**s discussed in Appendix A, the scope of this specification is limited to a description of how asset management tools can represent asset identification information when communicating it to other tools. It is out of scope of this specification to recommend which identifiers to use or to require that identification information be collected in a certain way or from a certain place. Higher-level specifications, tools, and organizations that implement Asset Identification, however, are encouraged to make these recommendations or specify these requirements in order to support the particular needs of their use cases.

5.2 Data Model Overview

The Asset Identification data model consists of a set of asset types and a set of information that can be provided about each asset type. The asset types currently supported in this specification are:

- Person
- Organization
- System
- Software
- Database
- Network
- Service
- Data
- Computing Device
- Circuit
- Website

The specification MAY be extended by Asset Identification producers to allow for other asset types as needed; however, it is OPTIONAL for Asset Identification consumers to support asset types not present in the core specification.

For each asset type above, the specification has a core set of fields that may be provided in order to identify an asset of that type. For example, an asset of type “person” may be identified by an email address, full name, telephone number, or birth date. Any number of these fields may be populated in order to create an asset identification element. Specifications, management environments, organizations, and tool vendors implementing Asset Identification are encouraged to recommend, restrict, or require that certain fields be populated or not populated; however, the specification itself does not do so.

There are several different types of information that may be used to identify assets: literal identifiers, relationship identifiers, synthetic identifiers, and extension identifiers. These four identifier types are differentiated only because they are represented differently in the data model. No identifier type is intrinsically more or less valuable for performing asset identification than any other identifier type.

5.2.1 Literal Identifiers

Literal identifiers are the pre-defined fields containing literal values that may identify an asset. For example, Media Access Control (MAC) address is an example of a literal identifier for a computing device. Literal identifiers defined in Asset Identification MUST be properly processed without error by Asset Identification consumers.

5.2.2 Synthetic Identifiers

Synthetic identifiers are meant to be used when a database or process assigns an identifier. For example, an employee is often assigned an employee identifier which may be used to track him or her across the organization. These identifiers should be represented using the synthetic identifier construct: the namespace denotes the management domain for which the identifier is valid and the identifier contains the identifier itself. Each asset type allows for a list of zero to many synthetic identifiers. Synthetic identifiers MUST be properly processed without error by Asset Identification consumers.

5.2.3 Relationship Identifiers

Relationship identifiers are meant to be used when an asset may be identified based on a relationship to another asset. For example, a system may be identified based on the fact that it is named “System 1” and it is connected to network “INTERNAL”. Relationship types are represented as a controlled vocabulary. Any relationships defined in the Asset Identification controlled vocabulary are core and MUST be processed without error by Asset Identification consumers. Relationships that are defined in other controlled vocabularies are considered extension identifiers and MAY be supported by Asset Identification consumers.

5.2.4 Extension Identifiers

Although this specification intends to support the most common types of information that are used to identify assets, certain users, organizations, or use cases may find that the core model does not support some fields that they need. Asset Identification supports a producer’s ability to provide these identifiers in any asset identification element through extension identifiers; however, it is OPTIONAL for consumers to process or understand these identifiers unless some other specification requires it. Extension identifiers may include additional literal values as well as relationship identifiers that are outside of the Asset Identification controlled vocabulary. Extension identifiers MUST be processed without error by consumers; however, consumers are encouraged to ignore identifying information that they do not understand and is not defined in the core schema in order to avoid false positives or false negatives.

5.3 Providing Asset Identifications

In the absence of other guidance or requirements, Asset Identification providers SHOULD provide as much information as they have available in the core (non-extension) asset identification element. Bandwidth constraints, other specifications, and tool intelligence MAY help define how much or which information SHOULD be provided beyond this recommendation.

5.4 Consuming Asset Identifications

Asset Identification consumers MUST be able to process literal identifiers, synthetic identifiers, relationship identifiers, and extension identifiers without error. In this context, “process” simply means ingest without error and optionally use as an input in performing a matching. Asset Identification consumers SHOULD process literal identifiers, synthetic identifiers, and relationship identifiers and support incorporating them into a matching process. Asset Identification consumers SHOULD NOT incorporate unknown extended identifiers into a matching process as they may be misleading or misunderstood.

Extended identifiers that are defined in another specification or policy that the consumer implements MAY be supported as appropriate and as defined by that specification or policy.

5.5 Matching

Matching is the process of determining whether or not two or more asset identification elements are referring to the same asset. Matching is performed across an entire asset identification element, not across each individual property of an identifier.

Although matching identifiers is an important part of the asset identification process, due to a wide variety of current tool practices, organizational architectures, and the need to allow for innovation, this specification does not provide any normative requirements in regards to matching. Tools are free to perform matching based on their own logic, and specifications implementing asset identification are encouraged to provide their own recommendations or requirements around matching.

5.6 Sample Correlation Workflow

The diagram in Figure 5-1 shows a sample correlation workflow, including matching discoverable information and several synthetic identifiers.

In this sample architecture, which is merely one example, several tools report on information about an asset. The information is correlated by an asset database, potentially processed or aggregated, and then reported to a higher-level database.

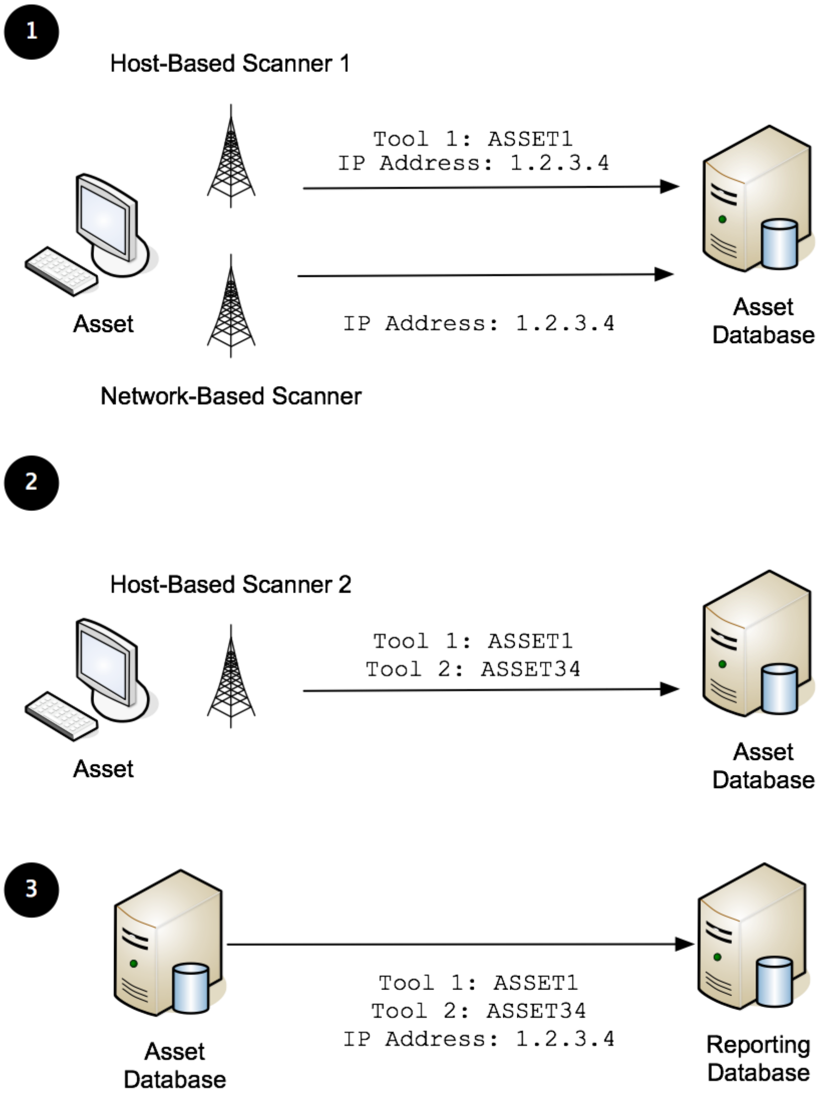

Figure 5-1: Sample Correlation Workflow

In step 1, a host-based scanner is reporting on asset information (e.g. vulnerability assessment results) using a synthetic identifier in its own namespace and an IP address as identifying information. Additionally, a network scanner is reporting on network events by IP address. This allows the asset database to correlate information coming from the network with information on the host, potentially matching vulnerabilities discovered by the host-based scanner with attacks against that vulnerability discovered by the network scanner.

In step 2, another host-based scanner (e.g., an asset inventory tool) reports data using both a synthetic identifier in its own namespace and a synthetic identifier in the first host-based tool's namespace. This other identifier may have been collected on the system or may have been discovered some other way; how that collection happens is out of scope of this specification. By passing both identifiers, however, the scanner provides enough data to the asset database to correlate the additional inventory data with the vulnerability and event data. Additionally, any other data reported using either the IP address or the two synthetic identifiers will be able to be correlated as well. Note that this specification does not require or recommend that tools process identifiers in certain ways: for example, an IP address may become stale after a period of time, but it is out of scope for this specification to recommend how to deal with that.

In step 3, the asset database provides a report to a higher-level database. Depending on the reporting architecture, organizational hierarchy, and reporting requirements, asset databases may want to report on data with different sets of synthetic identifiers for each asset. This specification does not restrict or recommend architectures or workflows.

In the reporting architecture shown above. all known asset identifiers for each asset are being used to report information to the higher level. In this architecture, data provided by other tools to the higher-level database may be correlated with the reported data. Other options would be to only report some information to the higher-level database or even to generate a new synthetic identifier to perform reporting, depending on the reporting requirements and network architectures.

6. Data Model

This section documents the data model for asset identification. The XML Schema that implements this data model is provided separate from this specification.

The asset identification model is a fairly flat model that defines the constructs to hold identifying information about an asset. A limited number of asset types are defined for which model constructs exist. In order to use the asset identification model,

- The user MUST produce an XML ai:assets or ai:asset-related element consistent with the data model described in Section 6.4.1
- The XML element produced MUST validate against the XML Schema (XSD) for Asset Identification 1.1 at http://scap.nist.gov/specifications/ai/index.html. In situations where the XML Schema does not match the documented model in this specification, the XSD takes precedence.

The following tables formalize the logical data model. The data contained in the tables are requirements, and MUST be interpreted as follows.

- The "Element Name" field indicates the name for the entity being described
- The "Definition" field indicates the prose description of the text. The field MAY contain requirement words as indicated in RFC 2119.
- The "Inherits" field indicates that the element takes on all of the properties of the inherited element in addition to the properties defined for the element
- The "Properties" field is broken into four columns
    - The "Name" column indicates the name of a property that MAY or MUST be included in the described element in accordance with the cardinality indicated in the "Count" field
    - The "Type" column indicates the type of data that MUST be the value for the property. There are two categories of types: literal and element. A literal type will indicate the type of literal. The element type will reference the name of another element that defines the content for that property.
    - The "Count" column indicates the cardinality of the property within the element. The property MUST be included in the element in accordance with the cardinality. If a range is given, and "n" is the upper-bound of the range, then the upper limit is unbounded.
    - The "Definition" column defines the property in the context of the element. The field MAY contain requirement words as indicated in RFC 2119.

Each literal data element MAY have a "source" attribute associated with it. The source attribute is intended to capture the source of the information for that data element. The field SHALL be a string type, but the value of the field is left to the content producer. The value MAY include, but is not restricted to, a synthetic ID of the asset that sourced the information, another ID of the source, or a description of the source. See the XSD for additional clarity on the "source" attribute.

Each literal data element MAY have a "timestamp" attribute associated with it. If populated, the timestamp attribute indicates when the data was generated for that element.

6.1 Abstract Elements

The element described in this section is the top-level abstract element from which all asset elements derive.

Table 6-1: Element – ai:asset

Element Name: ai:asset				
Definition	The root element from which all other asset elements derive. This element does not represent any specific type of asset, and therefore should only be used as a base class for other, concrete, asset elements. Any element that claims to be an asset element compliant with this specification SHALL directly or indirectly inherit all of the attributes in this element. The asset element SHALL NOT be used directly in an asset identification instance because it is abstract.			
Properties	**Name**	**Type**	**Count**	**Definition**
	synthetic-id	element – synthetic-id	0-n	Holds the synthetic ID information for the asset.
	location	element - one of: location-point, location-region, location-address (type xal:AddressDetails)	0-n	Holds the location information where the asset resides. xal:AddressDetails is defined in [xAL].
	extended-information	element – any XML	0-1	Holds extension identifiers for the asset. The content can be any well-formed XML defined in a namespace other than the Asset Identification namespace.
	timestamp	literal – dateTime	0-1	The date and time when the information on the asset was generated.

Table 6-2: Element – ai:it-asset

Element Name: ai:it-asset	
Inherits	ai:asset
Definition	An abstract element that extends from the asset element. it-asset is a placeholder element to carry common attributes related to IT assets. For the current iteration of this specification no common attributes have been identified, but future iterations of the specification may contain common attributes for IT assets. All asset elements that are describing IT assets SHOULD extend from the it-asset element.

6.2 Concrete Asset Elements

The following elements describe the data elements for the asset types defined in this specification.

Table 6-3: Element – ai:circuit

Element Name: ai:circuit				
Inherits	ai:it-asset			
Definition	Captures identifying information about a circuit.			
Properties	**Name**	**Type**	**Count**	**Definition**
	circuit-name	literal – token	0-1	The name of the circuit being identified.

Table 6-4: Element – ai:computing-device

Element Name: ai:computing-device				
Inherits	ai:it-asset			
Definition	Captures identifying information about a computing device.			
Properties	**Name**	**Type**	**Count**	**Definition**
	distinguished-name	literal – token	0-1	The X.500 distinguished name of the computing device being identified.
	fqdn	literal – token	0-1	The fully-qualified domain name for the computing device being identified.
	cpe	literal – token	0-n	The Common Platform Enumeration name for the computing device being identified. This MUST be a CPE 2.2 string [CPE22] or CPE 2.3 URI [CPE23]. CPE Regex: [c][pP][eE]:/[AHOaho]?(:[A-Za-z0-9\._\-~%]*){0,6}
	connection	element – ai:connection	0-n	Information about a network interface on the computing device being identified.
	hostname	literal – token	0-1	The hostname of the computing device.
	smbios-id	literal - string	0-1	The System Management BIOS identifier for the computing device.
	motherboard-guid	literal - string	0-1	The motherboard globally unique identifier of the computing device.

Table 6-5: Element – ai:data

Element Name: ai:data	
Inherits	ai:asset
Definition	A generic element to describe any type of data. Since this element is generic it does not define any of its own properties, but instead relies solely on the properties inherited from asset.

Table 6-6: Element – ai:database

<table>
<tr><th colspan="5">Element Name: ai:database</th></tr>
<tr><td><b>Inherits</b></td><td colspan="4">ai:it-asset</td></tr>
<tr><td><b>Definition</b></td><td colspan="4">Captures identifying information about a database.</td></tr>
<tr><td><b>Properties</b></td><td><b>Name</b></td><td><b>Type</b></td><td><b>Count</b></td><td><b>Definition</b></td></tr>
<tr><td></td><td>instance-name</td><td>literal – token</td><td>0-1</td><td>The name of the database instance.</td></tr>
</table>

Table 6-7: Element – ai:network

<table>
<tr><th colspan="5">Element Name: ai:network</th></tr>
<tr><td><b>Inherits</b></td><td colspan="4">ai:it-asset</td></tr>
<tr><td><b>Definition</b></td><td colspan="4">Captures identifying information about a network.</td></tr>
<tr><td><b>Properties</b></td><td><b>Name</b></td><td><b>Type</b></td><td><b>Count</b></td><td><b>Definition</b></td></tr>
<tr><td></td><td>network-name</td><td>literal – normalizedString</td><td>0-1</td><td>The name of the network being identified.</td></tr>
<tr><td></td><td>ip-net-range-start</td><td>literal - token</td><td>0-1</td><td>The starting IP address for the range of IP addresses for the network being identified.
IPv4 Regex: ^([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))$
IPv6 Regex: ([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}</td></tr>
<tr><td></td><td>ip-net-range-end</td><td>literal - token</td><td>0-1</td><td>The ending IP address for the range of IP addresses for the network being identified.
IPv4 Regex: ^([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))$
IPv6 Regex: ([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}</td></tr>
<tr><td></td><td>cidr</td><td>literal – token</td><td>0-1</td><td>The Classless Inter-Domain Routing information for the network being identified.</td></tr>
</table>

Table 6-8: Element – ai:organization

Element Name: ai:organization				
Inherits	ai:asset			
Definition	Captures identifying information about an organization.			
Properties	**Name**	**Type**	**Count**	**Definition**
	xnl:OrganisationNameDetails	element – xnl:OrganisationNameDetails	0-1	The name of the organization being identified. See [xNL] for details on populating this element.
	website-url	literal - URL	0-n	A website associated with the organization being identified.
	telephone-number	literal - token	0-n	A phone number associated with the organization being identified. For a North American number, the number must be valid and the format must be XXX-XXX-XXXX where X is a digit. For an international number, the number must begin with a '+' symbol, followed by 7 to 15 digits. A space may be used between digits, as appropriate. For example: +88 888 888 8 (this is following the ITU-T E.123 notation). Regex: ^(([2-9][0-8]\d-[2-9]\d{2}-[0-9]{4})|(\+([0-9] ?){6,14}[0-9]))$
	email-address	literal – token	0-n	An email address associated with the organization being identified.

Table 6-9: Element – ai:person

Element Name: ai:person				
Inherits	ai:asset			
Definition	Captures identifying information about a person.			
Properties	**Name**	**Type**	**Count**	**Definition**
	xnl:PersonName	element - xnl:PersonName	0-1	The name of the person being identified. The element type is defined in [xNL] and SHALL be used as documented in that specification.
	birth-date	literal - date	0-1	The birth date of the person being identified.

<table>
<tr><td></td><td>telephone-number</td><td>literal - token</td><td>0-n</td><td>A phone number associated with the person being identified. For a North American number, the number must be valid and the format must be XXX-XXX-XXXX where X is a digit. For an international number, the number must begin with a '+' symbol, followed by 7 to 15 digits. A space may be used between digits, as appropriate. For example: +88 888 888 8 (this is following the ITU-T E.123 notation).
Regex: ^(([2-9][0-8]\d-[2-9]\d{2}-[0-9]{4})|(\+([0-9] ?){6,14}[0-9]))$</td></tr>
<tr><td></td><td>email-address</td><td>literal – token</td><td>0-n</td><td>An email address associated with the person being identified.</td></tr>
</table>

Table 6-10: Element – ai:service

<table>
<tr><td colspan="5"><b>Element Name: ai:service</b></td></tr>
<tr><td><b>Inherits</b></td><td colspan="4">ai:it-asset</td></tr>
<tr><td><b>Definition</b></td><td colspan="4">Captures identifying information about a service running on a computing-device.</td></tr>
<tr><td><b>Properties</b></td><td><b>Name</b></td><td><b>Type</b></td><td><b>Count</b></td><td><b>Definition</b></td></tr>
<tr><td></td><td>host</td><td>literal – token</td><td>0-1</td><td>The IP address or fully qualified domain name of the host of the service.</td></tr>
<tr><td></td><td>ip-address</td><td>literal - token</td><td>0-1</td><td>The IP address that the service is running on.
IPv4 Regex: ^([0-9]||[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]||[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]||[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]||[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))$
IPv6 Regex: ([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}</td></tr>
<tr><td></td><td>port</td><td>literal – integer</td><td>0-n</td><td>The port number that the service is bound to.</td></tr>
<tr><td></td><td>port-range</td><td>literal – integer list</td><td>0-n</td><td>The lower and upper bound (inclusive) of the range of ports the service is bound to.</td></tr>
</table>

Table 6-11: Element – ai:software

<table>
<tr><td colspan="5"><b>Element Name: ai:software</b></td></tr>
<tr><td><b>Inherits</b></td><td colspan="4">ai:it-asset</td></tr>
<tr><td><b>Definition</b></td><td colspan="4">Captures identifying information about a class of software or a software instance.</td></tr>
<tr><td><b>Properties</b></td><td><b>Name</b></td><td><b>Type</b></td><td><b>Count</b></td><td><b>Definition</b></td></tr>
<tr><td></td><td>installation-id</td><td>literal – token</td><td>0-1</td><td>Any identifier for a software instance (installation). Use when identifying an instance of software and not just the class of software.</td></tr>
</table>

	cpe	literal - token	0-1	The Common Platform Enumeration name for the class of software being identified. This MUST be a CPE 2.2 string [CPE22] or CPE 2.3 URI [CPE23]. CPE Regex: [c][pP][eE]:/[AHOaho]?(:[A-Za-z0-9\._\-~%]*){0,6}
	license	literal – string	0-n	The license key associated with the software instance (installation). Use when identifying an instance of software and not just the class of software.

Table 6-12: Element – ai:system

Element Name: ai:system				
Inherits	ai:it-asset			
Definition	Captures identifying information about a system.			
Properties	**Name**	**Type**	**Count**	**Definition**
	system-name	literal – token	0-n	The name of the system being identified. This property can be replicated as systems may have multiple, or abbreviated, names. All of the names (including acronyms) may be captured here.
	version	literal - token	0-1	The version of the system being identified.

Table 6-13: Element – ai:website

Element Name: ai:website				
Inherits	ai:it-asset			
Definition	Captures identifying information about a website.			
Properties	**Name**	**Type**	**Count**	**Definition**
	document-root	literal - token	0-1	The absolute path to the document root location of the website on the host.

6.3 Helper Elements

Table 6-14: Element – ai:synthetic-id

Element Name: ai:synthetic-id				
Definition	Holds the synthetic identifier information for an asset.			
Properties	**Name**	**Type**	**Count**	**Definition**
	resource	literal - URI	1	A URI for the namespace in which the identifier is governed and unique.
	id	literal - token	1	The unique identifier for the asset within the resource namespace.

Table 6-15: Element – ai:connection

<table>
<tr><th colspan="5">Element Name: ai:connection</th></tr>
<tr><td><b>Definition</b></td><td colspan="4">Contains information relevant to a single connection to a network.</td></tr>
<tr><td><b>Properties</b></td><td><b>Name</b></td><td><b>Type</b></td><td><b>Count</b></td><td><b>Definition</b></td></tr>
<tr><td></td><td>ip-address</td><td>literal - token</td><td>0-1</td><td>The IP address for the connection.
IPv4 Regex: ^([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))$
IPv6 Regex: ([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}</td></tr>
<tr><td></td><td>mac-address</td><td>literal – token</td><td>0-1</td><td>The Media Access Control address for the network interface.
Regex: ([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2}</td></tr>
<tr><td></td><td>url</td><td>literal – URL</td><td>0-n</td><td>A Universal Resource Locator address for the network interface.</td></tr>
<tr><td></td><td>subnet-mask</td><td>literal – token</td><td>0-1</td><td>The subnet mask for the connection.
IPv4 Regex: ^([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))$
IPv6 Regex: ([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}</td></tr>
<tr><td></td><td>default-route</td><td>literal – token</td><td>0-1</td><td>The IP address for the default gateway for the connection.
IPv4 Regex: ^([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))\.([0-9]|[1-9][0-9]|1([0-9][0-9])|2([0-4][0-9]|5[0-5]))$
IPv6 Regex: ([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}</td></tr>
</table>

Table 6-16: Element – ai:location-point

Element Name: ai:location-point				
Definition	Contains a geographic coordinate system point			
Properties	**Name**	**Type**	**Count**	**Definition**
	latitude	literal – number	1	The latitude of the point represented as a number between -90 and 90. 90 = 90°N, -90 = 90°S. Value constraint: -90 <= x <= 90

	longitude	literal – number	1	The longitude of the point represented as a number between -180 and 180. 180 = 180°E, -180 = 180°W. Value constraint: -180 < x <= 180
	elevation	literal – number	0-1	The elevation of the point represented in meters above sea level. A negative number would indicate below sea level.
	radius	literal – number	0-1	The radius of a horizontal circle centered on the point within which the asset resides. Value constraint: x >= 0

Table 6-17: Element – ai:location-region

Element Name: ai:location-region				
Definition	Contains region information			
Properties	**Name**	**Type**	**Count**	**Definition**
	region-name	literal – token	1	The name of the region.

6.4 Relating Assets to Other Assets

While the assets modeled in Section 0, and their related elements, capture the literal values helpful for identifying the respective assets, it is often useful or necessary to define one or more relationships between assets. Those relationships can give additional context to the identifying algorithm in the implementing tool.

The Asset Identification data model allows for explicit relationships to be defined between an asset and one or more other assets. Each relationship is defined as {subject} {predicate} {object}, where {subject} is the asset from which the relationship begins, {predicate} is the relationship type being established, and {object} is one or more other assets. The predicate MUST be a qualified name that refers to a term in a controlled vocabulary. Section 6.4.1 documents the data model to represent assets along with relationships. Section 6.4.2 defines terms in a controlled vocabulary for Asset Identification.

6.4.1 Relationship Data Model

Asset Identification defines two elements that can be leveraged by specifications desiring to represent Asset Identification information. The first element, ai:asset-related, SHOULD be leveraged when the implementing specification desires to identify a single asset while demonstrating relationships between that asset and other assets. The second element, ai:assets, SHOULD be leveraged when the implementing specification desires to identify multiple assets while documenting the relationships between those assets and other assets. The two elements are documented in the following tables.

Table 6-18: Element – ai:asset-related

Element Name: ai:asset-related				
Definition	Identifies a single asset while capturing the relationships between that asset and other assets.			
Properties	**Name**	**Type**	**Count**	**Definition**
	asset-ref	literal – NCName	1	Contains the ID value of an ai:asset found on this ai:asset-related element. The asset referenced from this property is the primary asset of this element and SHALL be understood to be the asset being identified by this ai:asset-related element.
	relationships	element – core:relationships	0-1	Contains the relationships between assets identified in this element.
	asset	element – ai:asset	1-n	The assets captured in this element. This includes at minimum the primary asset referenced in asset-ref, as well as any additional assets that the primary asset is related to through a relationship.

Table 6-19: Element – ai:assets

Element Name: ai:assets				
Definition	Identifies multiple assets as well as the relationships between the assets.			
Properties	**Name**	**Type**	**Count**	**Definition**
	relationships	element – core:relationships	0-1	Contains the relationships between assets identified in this element.
	asset	element – ai:asset	1-n	The assets captured in this element.

Table 6-20: Element – core:relationships

Element Name: core:relationships				
Definition	Contains a collection of relationships between the report content and assets, report requests, and other reports.			
Properties	**Name**	**Type**	**Count**	**Definition**
	relationship	element – core:relationship	1-n	Contains a relationship between the subject and object(s) assets.

Table 6-21: Element – core:relationship

Element Name: core:relationship				
Definition	Contains a relationship between the subject and object(s) assets.			
Properties	**Name**	**Type**	**Count**	**Definition**
	type	literal - QName	1	This element contains the type of relationship that is being specified. The QName MUST refer to a term in a controlled vocabulary. The controlled vocabulary is identified by the namespace URI of the QName, and the term in that controlled vocabulary is specified by the local name of the QName. It is helpful, though not required, that when the namespace URI and local name are concatenated, the resulting URI is dereferenceable and points to a location that defines the term.
	scope	literal - token	0-1	Determines how to interpret multiple ref elements in a relationship. If used, this element MUST contain the string "inclusive" or "exclusive". When this element is not provided, its default value is "inclusive". When "inclusive" is specified, this relationship should be understood to exist between the subject asset and the collection of objects identified by the ref elements on this relationship. When "exclusive" is specified, this relationship should be understood to exist between the subject asset and each object asset identified by the ref elements individually.
	subject	literal – NCName	1	The property MUST identify the subject of the relationship by specifying the ID of the asset. Depending on the type of relationship being asserted, there may be additional restrictions on which type of asset may be referenced, but that will be documented with the vocabulary term.
	ref	literal - NCName	1-n	This element MUST identify the object of this relationship by specifying the ID of the asset. Depending on the type of relationship being asserted, there may be additional restrictions on which types of objects may be referenced, but that will be documented with the vocabulary term.

6.4.2 Relationship Types

Defined below are terms in a controlled vocabulary for Asset Identification. It is OPTIONAL that content producers use the terms defined below, but all Asset Identification compliant implementations MUST understand the terms defined in this section. Content producers SHOULD use these terms when possible.

All terms listed in Table 6-22 exist in the controlled vocabulary identified by http://scap.nist.gov/specifications/ai/vocabulary/relationships/1.0#. The definition of each term can also be found at the URL created when concatenating the URL and the term together. The table MUST be interpreted as follows:

- The “Term” column indicates the local-name of the term being identified.
- The “Domain” column indicates the exhaustive set of subject types that may be referenced by a relationship of that type. A relationship of that type MUST reference a subject of the type indicated in “Domain” for that relationship.
- The “Range” column indicates the exhaustive set of object types that may be referenced by a relationship of that type. A relationship of that type MUST reference an object of the type indicated in “Range” for that relationship.
- The “Description” column contains a prose description of the relationship type. This column may contain requirement words as indicated in [RFC 2119]. Those requirement words MUST be interpreted as described in [RFC 2119] for the relationship.

Table 6-22: Controlled Vocabulary Defined for ARF

Term	Domain	Range	Description
hasTerminationDevice	ai:circuit	ai:computing-device	The circuit is terminated by the device.
hasServiceProvider	ai:circuit	ai:organization	The circuit is owner/operated by the organization.
hasNetworkTerminationPoint	ai:circuit	ai:network	The circuit ends at the network.
servedBy	ai:database, ai:website	ai:service	The database or website is served up by the service.
hasServiceProvider	ai:service	ai:software	The service is provided by the software.
installedOnDevice	ai:software	ai:computing-device	The software is installed on the computing device.
connectedToNetwork	ai:system	ai:network	The system is connected to the network.

Content producers that choose to use terms that are not listed in Table 6-22, or to use other terms in addition to those listed in Table 6-22, MAY do so while still remaining compliant to this specification. Content producers SHALL always use terms defined in a controlled vocabulary. The controlled vocabulary SHALL be identified using a URI. Concatenating the controlled vocabulary URI with a term in the vocabulary MAY create a dereferencable URI that points to a definition for that term. This is often accomplished by using an HTTP URL for the controlled vocabulary URI, and ending that URL in “#” or “/”. For instance, http://scap.nist.gov/specifications/ai/vocabulary/relationships/1.0#installedOnDevice is a deferenceable link to the definition of “installedOnDevice”.

Appendix A—Use Cases

The following use cases describe some common asset management processes that rely on the ability to uniquely identify assets. The asset identification specification was developed primarily in order to support these use cases, although the specification may be useful for other processes or uses.

A.1 Correlation of Sensed Data

Sensor data is not limited to an automated process: user surveys, manually entered information, and other data may also be correlated using this Asset Identification specification. The data must be correlated both with other manually collected data and with data collected by automated sensors, in order to build a complete representation of all known data about an asset.

Consistent asset identification allows data to be correlated regardless of:

- Collection timeframe
- Data type (vulnerability scan vs. user survey)
- Manual or automated process
- Data format

In this case, the goal of asset identification is to provide as much identifying information as possible about an asset in order to ensure the greatest probability of matching asset data from several different sources. Constraining which data is provided merely reduces the possibility of a match.

A.2 Federation of Asset Databases

While many smaller organizations may only have a single asset database, larger organizations with many asset databases may wish to share information about assets among them. This includes:

- Peer to peer relationships, where asset data is replicated and fused between several asset databases
- Hierarchical relationships, where an asset database is aggregated from several lower-level databases into fewer higher-level databases

In both use cases, asset identification facilitates detection of duplicate asset representations, correlation of asset data across the databases, and direct queries for asset data among tools.

In this use case, asset databases may wish to use a smaller set of data, or even a single identifier, in order to properly federate the asset data. For example, use of the motherboard GUID or single synthetic ID, such as an asset tag, may be sufficient to allow asset databases to exchange information about assets.

A.3 Directly Targeted Remediation Actions

While assessment and sensor data may be collected from all assets in a particular organization environment without specifically identifying a particular asset identifier, remediation actions require a more granular identification process that directly targets the asset using identification data. This ensures that unintended side effects are avoided and the intended remediation action is able to be completed successfully.

A single agent identifier or motherboard GUID allows those triggering remediation actions to specify exactly which assets should have the remediation applied and allows the tools to unambiguously identify those assets in the remediation control language.

A.4 Management of Asset Data

Outside of the collection of sensor data and federation of asset databases, asset data may be used in a variety of management processes. This includes both further processing of asset data, such as aggregation for the purposes of metrics collection, and display to an end user. Both of these uses require asset identification be present to ensure all systems are able to accurately represent the correct assets. For purposes of aggregation, for example, asset identification may be used to request detailed data about outliers from the sensors that collected the data.

Appendix B—Normative References

The following documents are indispensible references for understanding the application of this specification.

[CPE22] The MITRE Corporation (MITRE) Common Platform Enumeration (CPE) Specification Version 2.2, March 2009. See: http://cpe.mitre.org

[CPE23] NIST Interagency Report 7695, Common Platform Enumeration: Naming Specification Version 2.3 (Draft), August 2010. See: http://csrc.nist.gov/publications/PubsNISTIRs.html

[RFC 2119] Internet Engineering Task Force (IETF) Request for Comment (RFC) 2119: Key words for use in RFCs to Indicate Requirement Levels, March 1997. See: http://www.ietf.org/rfc/rfc2119.txt

[RFC 2396] Internet Engineering Task Force (IETF) Request for Comment (RFC) 2396: Uniform Resource Identifiers (URI): Generic Syntax, August 1998. See: http://www.ietf.org/rfc/rfc2396.txt

[xAL] Organization for the Advancement of Structured Information Standards (OASIS) Extensible Address Language (xAL) Version 2.0, 24 July 2002. See: http://www.oasis-open.org/committees/ciq/ciq.html#6

[XML] W3C Recommendation Extensible Markup Language (XML) 1.0 (Fifth Edition), 26 November 2008. See: http://www.w3.org/TR/REC-xml/

[XML Schema] W3C Recommendation XML Schema, 28 October 2004. See: http://www.w3.org/XML/Schema.html

[xNL] Organization for the Advancement of Structured Information Standards (OASIS) Extensible Name Language (xNL) Version 2.0, 24 July 2002. See: http://www.oasis-open.org/committees/ciq/ciq.html#5